INTERPRETIVE HEATBUGS: DESIGN AND IMPLEMENTATION

V.S. MELLARKOD,* Texas Tech University, Lubbock, TX,
and Argonne National Laboratory, Argonne, IL
D.L. SALLACH, Argonne National Laboratory, Argonne, IL,
and The University of Chicago, Chicago, IL

ABSTRACT

Agent-based modeling has been proven to be an effective strategy for expressing social behavior. In order to extend the veracity of social models and capture related phenomena, we introduce the concept of interpretive mechanisms in social agents. The central concept is to allow an agent to view others and the environment through his own interpretation of events and situations. There are three main mechanisms that we introduce [Sallach 2003] to capture interpretive behavior: prototype inference, social accounting, and situation definition. Agents use the mechanisms in their response cycles to infer situated meaning.

Our current work involves building a small interpretive application that can be used as a basic exemplar for the use of interpretive mechanisms in social modeling. Similar to the role played by heatbugs as a basic example for agent-based modeling; we introduce interpretive heatbugs (IHBs), which can be used as a first example for interpretive social agent modeling. This paper discusses the design and implementation strategies of interpretive heatbugs. The implementation is done in J programming language, which we are investigating as a potentially effective language for the exploration of interpretive agents.

Keywords: Agent-based models, interpretive agents, prototype inference, social accounting, situation definition, array programming

INTRODUCTION

Since antiquity, it has been assumed that the concepts employed by the human mind can be described in Aristotelian form (i.e., objects are organized by genus [class] coupled with differentia [distinguishing characteristics sufficient to produce an unambiguous definition]). A bird, for example, is sometimes defined as a biped (genus) with feathers (differentia).

In the final quarter of the twentieth century, however, cognitive science research called the Aristotelian model into question. Seminal and well-replicated studies reveal that human conceptual structures are organized in terms of family resemblances (Rosch 1978; Heit 1997). A prototype or exemplar defines a reference point, relative to which other examples are classified in terms of their similarity, along radial dimensions of difference. Stated differently, prototypes may be regarded as a focal object or event that serves as the reference point for objects or events that are more or less similar.

---

* *Corresponding author address*: Veena S. Mellarkod, Department of Computer Science, Texas Tech University, 2500 Broadway, Lubbock, TX 79409; email: mellarko@cs.ttu.edu.

An entire concept, referenced by its prototype, is the subject of proximity-based reasoning. Rosch (1983) refers to the overall cognitive process as reference point reasoning. The latter incorporates the typicality of any given instance relative to the radial structure of the concept as a whole. In a given situation, various prototypes may be comparatively assessed as to which one is the most appropriate for understanding the entity and/or situation at hand. When considered in conjunction with the Miller (1956) constant (an early formulation of bounded rationality constraints), the latter defines a (slightly variable) constraint that controls the number of prototypes considered in such comparisons.

The interpretive heat bugs (IHBs) application is a reference example for introducing interpretive mechanisms in agent-based modeling. This paper deals with the design and implementation of this reference application. Three mechanisms — prototype reasoning, situation definition, and orientation accounting — are the basis for introducing interpretive behavior in agents. IHBs note the behavior of their neighbors, and, on the basis of their observations (including the neighbors' ethnic and religious markers), they construct #nice, #similar, and derivative prototypes that can then be used in applying their distinctive rules to new situations. The following text briefly introduces an IHBs example; this is followed by a discussion of the design methodology.

INTERPRETIVE HEATBUGS

The IHBs concept consists of a world as an environment and bugs (agents) who live in this world. The world is a place where heat gets diffused across the surroundings according to standard heat diffusion laws (similar to heatbugs). The bugs are more complex than the regular heatbugs application; they have a religion and ethnicity. They also have religiosity and clusivity factors that portray the depth of their religious and ethnic beliefs, respectively. They have a level of aggressiveness and generosity that depend on their religious and ethnic beliefs. The agents output heat to the surroundings at constant periods of time. They also prefer a temperature zone at which they are comfortable. There are other temperature zones where they feel mildly uncomfortable (hot or cold), and at the rest of the temperatures, they are extremely uncomfortable, to the point of distress. The agents prefer being in their comfort zones; so they find the best possible neighborhood around them and try and move to that place. When confronted by another agent trying to get the same place, the agent analyzes the situation and determines whether it wants to shove others or not. The shove rules, which depend on the agent's religion and ethnicity, help him to decide. The shove rules make use of the situation that the agent currently perceives and depend on several prototypes, such as #nice and #similar. The strength of the shove depends on the agent's aggressiveness. The agent with maximum shove strength wins the place, and others get pushed backed to their original places. The agents also possess resources that they lose or gain depending on their zonal situation.

DESIGN OF IHB

The design of IHBs integrates the regular heatbug design and the response cycle of an agent. In particular, the parts that emit heat and diffuse heat are the same in both the applications. Apart from the regular initialization of parameters, IHB initialization also deals with the initialization of the prototypes the agents possess. There are two prototypes being considered for this application: #nice and #similar. Other prototypes, like #mean and #tough, can be incorporated while extending this application example to a real domain. The flowchart in Figure 1 describes the operational sequence of the application. The dotted line encompasses the step function, which is repeatedly executed to run the simulation.

At initialization, agents are randomly assigned a religion, subreligion, and ethnicity according to the distributions expressed. The initialization graphical user interface (GUI) allows a user to mention the parameters necessary for the simulation. The arbitrary religions and subreligions used are shown here:

Religion	Subreligions
AA	AA, BA, CA
BB	AB, BB, CB
CC	AC, BC, CC

The agents are also divided into different ethnicities — X, Y, and Z — where X implies exclusive, Z implies inclusive, and Y denotes no preference. The agents have resources in the nature of health that they could use.

The simulation proceeds as follows. At every step, agents emit heat, and the heat diffuses in the environment according to the standard heat diffusion laws. An agent finds a desirable position in the Moore neighborhood and would like to move there. When he takes a half step toward the desired position, he can see other agents (if any) who also want to move to the same position. The agent sees this as a new situation and would like to decide whether he still wants to pursue moving to the desirable position. There are several factors that define this situation: the agent's current position (i.e., whether he is comfortable, uncomfortable, or extremely uncomfortable); the competing agents that are involved; the agent's knowledge about these

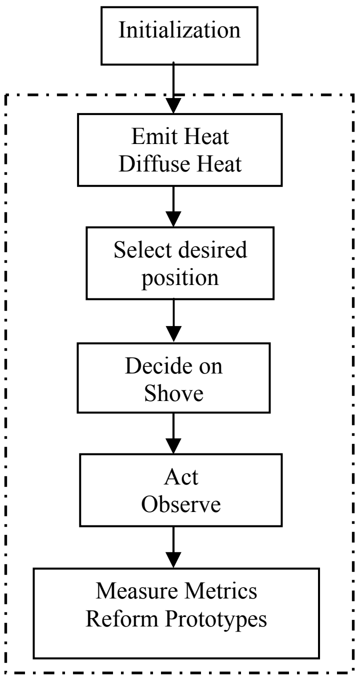


FIGURE 1 IHB flowchart

agents; and the agent's religion, subreligion, and ethnicity, which shape the agent's shove rules, which, in turn, govern the decision of whether to pursue the goal in the picture. The shove rules used are shown in Table 1. These rules use prototypes of the agents and their knowledge about the other agents that they acquire over time. After the agent analyses the situation and decides his particular response, he acts accordingly. Finally, agents observe others' actions, record them, and create/update their prototypes.

Initialization

Major parts of initialization deal with creating the initial prototypes with which the agents start. This is seen as parental knowledge contributed to the child about other ethnicities, religions, subreligions, etc. Each agent has two prototypes that need to be initialized. The #nice prototype consists of three dimensions: nz, ns, and nn. The #similar prototype consists of six dimensions: religion, subreligion, ethnicity, and sz, ss, and sn metrics. The dimensions or attributes associated with each prototype and their definitions are defined in section classification metrics. The definitions of some of these metrics use prototype clusters, which can, in turn, be generated only when the metric values are calculated. This recursive definition makes the initialization of prototypes nontrivial.

A sample set of agents is used to run the initialization. These agents undergo the same steps of simulation, with little variations, in order to build prototypes. Initialization is divided into three parts denoted as parts a, b, and c. Part a is very much like regular heatbugs with a little variation (i.e., an agent knows another agent's religion, subreligion, and ethnicity when he is in the agent's Moore neighborhood). This part is executed for an arbitrary number of times when the agents move around sequentially (as in heatbugs) and become familiar with their neighbors. Part b deals with the execution of shoving actions when an agent's requirements in shove rules get satisfied and his aggressiveness is greater than a particular level. These actions are observed by the other agents around. The metrics calculated are nz and sz, which do not require the clusters for calculation. Part b is executed for an arbitrary number of times, and the agents then have a collection of observed acts. These acts are used to find an initial similarity prototype that also uses the religion, subreligion, and ethnic dimensions. So after Part b is executed, similar prototype clusters are formed with dimensions: Religion, subreligion, ethnicity, sz metrics, and nice prototypes consist of only one dimension, which is nz. In Part c, the simulation is executed as described in the flowchart, and the other metrics that require prototype clusters use the partial clusters formed in Part b as their basis and build upon them. At the end of the initialization, the environment is again cooled, the bugs are again given arbitrary positions, and the prototype clusters that have been built are used for the agents as the initial seed.

Shove Rules

The shove rules are organized by bug ethnicity and religion. They are designed to not illustrate any essential qualities but instead to generate the diverse patterns of action, as well as uncertainty, regarding the relationship between the social structure and patterns of action. These patterns are intended to correspond to the complexities of naturally occurring cultures.

TABLE 1 IHB shove rules[a]

Prototype Shove Rules	XX		YY		ZZ	
	Rule 1	Rule 2	Rule 1	Rule 2	Rule 1	Rule 2
AAA Requisite Except	Distress None	Zone improve *#Nice #simEth*	Zone improve *#Nice* bug		Distress None	Zone improve CLUS >> CRV (& min *#simRel* & in pDistress)
AAB Requisite Except	Distress *#Nice #simEth*		Distress CLUS < CRV & *#nice #simEth* CLUS > CRV & *#nice* bug		Distress more *#nice* bugs	
AAC Requisite Except	Zone or large improve more *#nice #simEth*		Distress None	Large improve CLUS > CRV & *#nice* bug CLUS < CRV & *#nice #simEth*	Zone improve None	Large temp improve *#simRel* & in pDistress
BBA Requisite Except	Distress Very *#simBoth*		Distress *#nice* bug		Distress & all *#mean*	
BBB Requisite Except	Distress All *#simEth*		Distress CLUS >0 & *#nice* bug CLUS <0 & *#nice #simEth*		Distress CLUS >>0 & very *#simRel*	
BBC Requisite Except	Zone improve None	Temp improve More very *#simBoth*	Temp improve CLUS< 0 & (more *#simEth* & in pWorse zone)		Temp improv None	

TABLE 1 (Cont.)

Prototype Shove Rules	XX		YY		ZZ	
	Rule 1	Rule 2	Rule 1	Rule 2	Rule 1	Rule 2
CCA Requisite / Except	Zone improve / None	Temp improve / More very *#simBoth*	Zone improve / None	Temp improve / CLUS < 0 & all *#simEth*	Zone improve / None	Temp improve / Very *#simRel*
CCB Requisite / Except	Zone improve / More *#simEth*		Distress / None	Zone improve / CLUS < CRV & (*#simEth* & in pDistress) CLUS > CRV & (very *#simRel* & in pDistress)	Zone improve / *#nice* very *#simRel* & in pDistress	
CCC Requisite / Except	Zone or large improve / More *#simEth*		Zone or large improve / CLUS < CRV & more *#simEth* CLUS > CRV more very *#simRel*		Zone or large improve / CLUS >> CRV & very *#simRel*	

[a] pDistress and pWorse stand for “projected” distress and worse, respectively, meaning it is “distress” and “worse” from the decision-making bug’s perspective. CRV is the cluster reference value used to subdivide the bug category. In baseline IHB, CRV = 0. Italics indicate prototypes.

IHB rules have two parts: prerequisites and exceptions. Prerequisites concern heat conditions and heat tolerance. They are exogenously and stochastically defined by ethnic and religious groups, but they are also distinctly determined for each bug. Exceptions address how the bugs recognize other bugs, particularly with regard to what categories of bugs are (or are not) taken into account in the decision to shove (or not shove). Exceptions rely heavily on #similar and #nice prototypes and their derivatives (e.g., #mean, #simEth, #simRel).

Prototypes are calculated by using hierarchical agglomerative cluster analysis using a Euclidean distance metric. Bugs classify a baseline of cases during initialization (analogous to parental instruction) and continue to observe and classify the actions of other bugs during the simulation run. These clusters form the basis of the bug's prototypes and are used to uniquely activate the bug's shove rules.

Classification Metrics

A bug's prototype clusters represent the bug's notion/idea/concept of other bugs. This is calculated by observing the actions of other bugs. A bug can observe the actions of others in two ways: as an observer who actively looks at a desired cell and the situation in that cell or as an actor who acts (by shoving, etc.) for a particular cell and observes others' actions related to that cell. Let us refer to a specific bug (either an observer or an actor) as a "self" that is actively classifying the actions of other bugs of interest and creating a world of its own thoughts, interpretations, and conclusions about the others. This bug's prototype clusters are these concepts. The two prototypes being dealt with here are #similar and #nice. The others of interest are named as neighbors.[1]

The self observes the actions of the neighbors and calculates some metrics as a means of interpreting their actions. The action of a neighbor is toward other neighbors (which may or may not include self). While calculating the metrics, self looks at a neighbor's action with respect to himself and others affected by the action. (Thus, there may be multiple actors in a situation.) Each acting neighbor is analyzed separately, and metrics for each one are calculated. During a given assessment, the neighbor being observed is in the role of an actor, while others in the same observation can be seen as reactors. While the action of a neighbor is being assessed by self, others are reactors. The resulting metrics can be sophisticated and/or widely divergent. We use two types of simple metrics to define the actions: #nice and #similar. The next few paragraphs define the current metrics.

Nice Metrics

The nice clusters are defined by using a set of three metrics: nz, ns, nn. The following discussion considers each of them in detail. We start with the nz metric. It captures an actor's niceness as shown toward his neighbors in extreme temperatures. Sympathetic actions of an actor are given more recognition in extreme heat zones than in comfortable zones. More precisely, the nz metric says, "Not shoving another in extreme situations is nicer than not shoving another in more comfortable situations." The degree of niceness is measured by using an nz metric table. The action of an actor with respect to others' situations is given a niceness measure between –1 and 1 by using Table 2.

This is the nz metric related to the not-shove action. The value in the first column and bottom row of 0.9 can be read as follows: "The actor who did not shove another in order to move from a distress situation to a comfortable situation is 0.9 nice according to the nz metric." A similar table representing metrics for the shove action is shown in Table 3. As mentioned earlier, these metrics can be as complex as necessary. Here, for the sake of clarity, we chose to keep it simple. It should be noted that the operative definitions are perspectival. For example, the self

[1] One point to note is that these "neighbors" do not need to be self's Moore neighbors.

TABLE 2 Not-shove niceness metric

Not Shove From ↓ To →	Comfortable	Uncomfortable	Distress
Comfortable	0.2	0.15	0.1
Uncomfortable	0.6	0.5	0.4
Distress	0.9	0.6	0.2

TABLE 3 Shove niceness metric

Shove From ↓ To →	Comfortable	Uncomfortable	Distress
Comfortable	-0.8	-0.85	-0.9
Uncomfortable	-0.4	-0.5	-0.6
Distress	-0.1	-0.4	-0.8

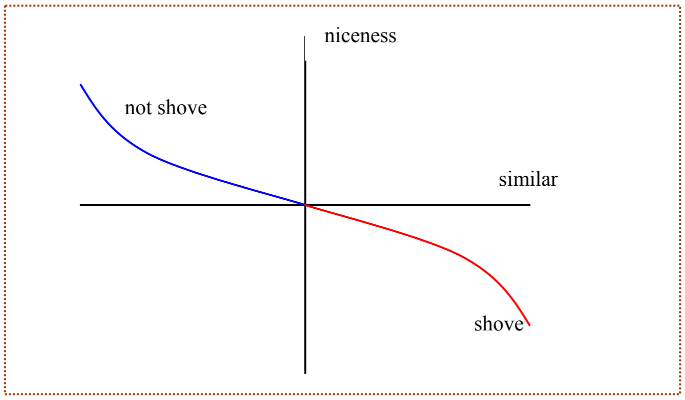
sees an actor to be in a distress situation and wants to move to a comfortable situation. The actor, on the other hand, may have a different definition of distress and thus might be comfortable in his present position and hence not want to shove.

The ns metric deals with how actors behave toward similar bugs. These statements apply: "Not shoving a dissimilar bug is nicer than not shoving a similar bug," and "Shoving a similar bug is regarded as being less nice." The metric is approximated in Figure 2.

Finally, the nn metric deals with how actors behave toward nice bugs. "Not shoving a bug that is less nice is nicer than not shoving a bug that is more nice," since not shoving a nicer bug is expected. This metric is shown in Figure 3.

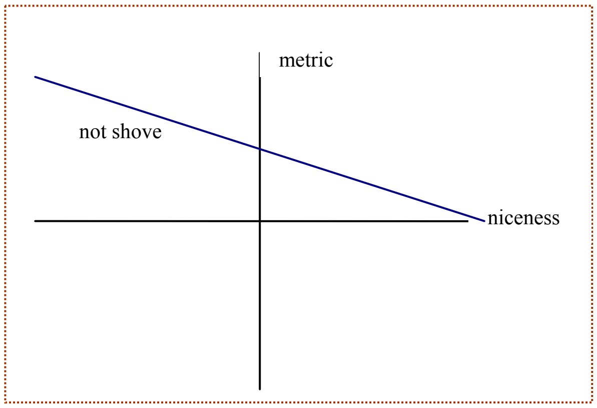
Similarity Metrics

Apart from religion, subreligion, and ethnicity factors, the following three metrics are used to define the similarity clusters: sz, ss, and sn. The sz metric represents the similarity in dealing with the situation. The self looks at the action of the actor and its situation (zone and reactors) and decides whether it (self) would act in the same way. The similarity/dissimilarity in action is represented by using a metric table called the difference metric. The ss metric represents the current similarity of the actor to self as seen by self. The metric is calculated as SCDsa, which is the similarity cluster distance between self and the actor. The sn metric represents the similarity of the actor with respect to self in niceness level. It is calculated as NCDsa, which is the nice cluster distance between self and actor.



Average $[(SC_{dan} - SC_{davg}) * I * \alpha_s] / SC_{davg}$.
For shove: $I = 0$ if $(SC_{dan} - SC_{davg}) > 0$ and $I = 1$ if otherwise.
For not shove: $I = 1$ if $(SC_{dan} - SC_{davg}) > 0$ and $I = 0$ if otherwise. SC_{dan} is the similar cluster distance between the actor and reactor, SC_{davg} is the cluster average distance of the similar clusters of self, I is an indicator function defined as above, and α_s is the salience given by self to this metric and is set to 1 as the default.

FIGURE 2 Notional interaction of two prototypes



The metric is calculated as Average (NCDnv), where NCDnv is the nice cluster distance between the acceptor and self's view of very nice cluster.

FIGURE 3 Relative niceness metric

PROTOTYPE CLUSTERS

The actions executed by the neighbors are recorded by proximate observers in two tables: events and event-members. These tables are then used to calculate the metrics defined above. There are five values recorded: from-zone, to-zone, actor, action, and event-num. The from-zone and to-zone are the actor’s current position and desired position (for which he shoves or not) as seen by the observer/self. The action is shove or not-shove. The event-num keeps track of all the actors in a particular event or situation that the self observes. This is used while calculating the metrics: to recognize the actors and reactors. The metrics are then used to form the prototype clusters.

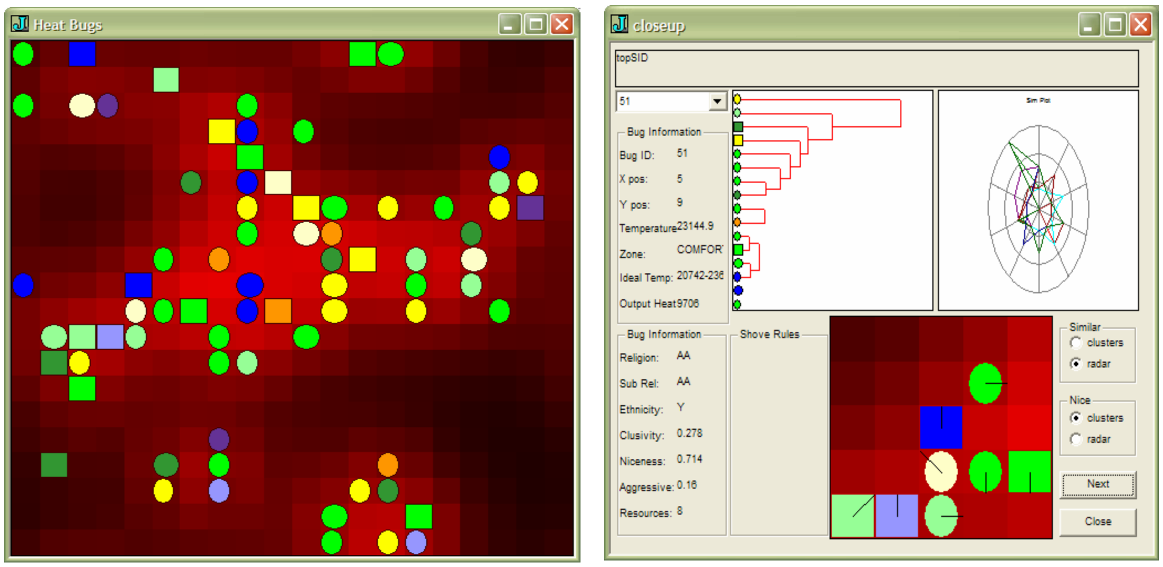
A random selection within the Miller magic number range (7± 2) is used to form the clusters. The cluster algorithm used is hierarchical agglomerative clustering. The clusters thus formed are used in shove rules. For example, the shove rule 2 for AAAA and X bug says, “If the move gives a zone improvement, then shove, unless there is a #nice and #similar bug with the same ethnicity.” Thus, if self finds itself in a situation where there are bugs competing for the desirable position, it will shove unless it recognizes one of the competing bugs as a nice and similar bug with ethnicity X. A bug is considered nice if it is in the very #nice cluster or the next-closest cluster to it, as attributed by the self. A bug is considered similar if the self and the bug are members of the same or adjacent #similar clusters. A fast join, which is the intersection of the two clusters, is performed. If a competing bug is a member of this similar-nice joining, then the self does not shove and remains in its place.

IHB IMPLEMENTATION

IHB implementation was done in an array programming language called J (Thomson 2001; Peele 2005). J is a mathematical language containing high-level primitives useful for building complex programs in fewer lines of code. This implementation was also a test for experimenting with using J as a language for agent-based modeling and simulation.

The initial user interface allows the user to change some of the parameters for the simulation. The heatbug interface showing the heatbugs and the environment looks like a standard interface of heatbugs. It is shown in Figure 4a. The colors of the bugs depend on their religions and subreligions. The greens are AAs, with three shades of green for three subreligions. The yellows are BBs, and the blues are CCs. Lighter and darker shades are given for each color to show three subreligions in each group. The shapes define the ethnicity to which the bugs belong. The Xs, who tend to be exclusive, are rectangles; the Zs, who tend to be inclusive, are circles; and the Ys, who are neither, are oval.

The resources of the bugs increase when they are comfortable and decrease when they are in distress. These resources combine with the bugs’ aggressiveness factor to give strength to their shoves. The bug with the highest strength in shoving wins the position, and the resources get depleted by the amount used. If there is a tie, a random contender wins the position. If all the agents decide not to shove, then none of them move into the position.



51

Bug Information

Bug ID: 51
X pos: 5
Y pos: 9
Temperature 23144.9
Zone: COMFOR
Ideal Temp: 20742-236
Output Heat 9706

Bug Information

Religion: AA
Sub Rel: AA
Ethnicity: Y
Clusivity: 0.278
Niceness: 0.714
Aggressive: 0.16
Resources: 8

FIGURE 4 Interpretive heatbugs (a) and CloseUp interfaces (b)

Interaction Walkthrough

The interpretive agent's interaction with the environment is fairly complex. Because it is difficult to understand the mechanisms in operation, it is useful to get a close view of interesting interactions and their effects. A CloseUp interaction walkthrough visualization was designed to work toward this goal. A snapshot of the interface is shown in Figure 4b. The CloseUp can be opened by double-clicking any bug on the main window. The bugs in the extended Moore neighborhood of range two are shown in the lower window. Each of the bugs in this CloseUp can be analyzed by looking at his internal properties and his nice and similar clusters, or the radar plots of their dimensions can be displayed. The shove rules are shown, and a user can walk through the four steps involved in the whole process by clicking the next button. In the first step, the bugs emit heat, and the heat diffuses. In the second step, the desired neighboring Moore cell is found. The third step presents a new situation to the bugs, wherein they can see who the competitors who are aspiring for the same position are, and they decide whether to use force to get the position. The fourth step implements the actions, and the bugs observe the situation from their viewpoints and re-categorize their clusters accordingly. Using the CloseUp interface, a user can analyze the microinteractions among the agents. Each agent in the CloseUp can be selected from the drop-down menu, and all the bug's attributes, including its current prototype definitions, are displayed. We believe that such interfaces will, in general, help in understanding details at the microlevel and thus help in analyzing emergent behaviors.

CONCLUSIONS

Interpretive agents in agent-based modeling represent a new area of research that we believe has the potential to decrease the grain and increase the veracity of social models, thereby increasing the potential for representing nonlinear and other phenomena of interest. We introduce a reference application as a resource for interpretive agent research and describe the design and implementation of this application.

ACKNOWLEDGMENT

This work is sponsored by the U.S. Department of Energy, Office of Science, under contract W-31-109-Eng-38.

REFERENCES

Blau, P.M. 1977. *Inequality and Heterogeneity: A Primitive Theory of Social Structure.* New York, NY: Free Press.

Collins, R. 1981. “Three faces of cruelty: Toward a comparative sociology of violence,” pp. 133–158 in *Sociology Since Midcentury: Essays in Theory Accumulation.* New York, NY: Academic Press.

Collins, R. 1987. “Interaction ritual chains, power and property: The micro-macro connection as an empirically based theoretical problem,” pp. 193–206 in J.C. Alexander, B. Giesen, R. Munch, and N.J. Smelser, eds., *The Micro-Macro Link.* Berkeley, CA: University of California Press.

Collins, R. 1990. “Stratification, emotional energy and the transient emotions,” pp. 27–57 in T.D. Kemper, ed., *Research Agendas in the Sociology of Emotions*. Albany, NY: SUNY Press.

Collins, R. 2000. “Situational stratification: A micro-macro theory of inequality,” *Sociological Theory* 18:17–43.

Heit, E. 1997. “Knowledge and concept learning,” in K. Lamberts and D. Shanks, eds., *Knowledge Concepts and Categories*. Cambridge, MA: MIT Press.

Joas, H. 1996. *The Creativity of Action.* Chicago, IL: The University of Chicago Press.

Jurgens, M. 2002. *Index Structures for Data Warehouses.* New York. NY: Springer.

Mellarkod, V.S., and D.L. Sallach. 2005. “Meaning-oriented social interactions: Conceptual prototypes, inferences and blends,” presented at the Lake Arrowhead Conference on Human Complex Systems.

Miller, G.A. 1956. “The magical number seven, plus or minus two: Some limits on our capacity for processing information,” *Psychological Review 63*:81–97.

Peele, H. 2005. *Mathematical Computing in J: Introduction.* Hertfordshire, UK: Research Studies Press.

Rosch, E. 1978. “Principles of categorization,” in E. Rosch and B.B. Lloyd, eds., *Cognition and Categorization.* Hillsdale, NJ: Lawrence Erlbaum.

Rosch, E. 1983. “Prototype classification and logical classification,” in E.K. Scholnick, ed., *New Trends in Conceptual Representation: Challenges to Piaget’s Theory?* Hillsdale, NJ: Lawrence Erlbaum.

Sallach, D.L. 2000. “Classical social processes: Attractor and computational models,” *Journal of Mathematical Sociology 24*:245–272 (winter).

Sallach, D.L. 2003. “Interpretive agents: Identifying principles, designing mechanisms,” in *Proceedings of the Agent 2003 Conference on Challenges in Social Simulation,* presented by Argonne National Laboratory and The University of Chicago, Oct. 2–4.

Sallach, D.L., and V.S. Mellarkod. 2004. “Prototype inference and social interaction,” in C.M. Macal, D. Sallach, and M.J. North (eds.), *Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence,* ANL/DIS-05-6, co-sponsored by The University of Chicago and Argonne National Laboratory, Oct. 7–9.

Sallach, D.L., and V.S. Mellarkod. 2005. “Accounting for agent orientations,” presented at the Lake Arrowhead Conference on Human Complex Systems.

Sallach, D.L., and V.S. Mellarkod. 2005. “Interpretive agents: A heatbug reference simulation,” in C.M. Macal, D. Sallach, and M.J. North (eds.), *Proceedings of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms,* ANL/DIS-06-1, co-sponsored by Argonne National Laboratory and The University of Chicago, Oct. 13-15.

Scheff, T.J. 1990. *Microsociology: Discourse, Emotion and Social Structure.* Chicago, IL: The University of Chicago Press.

Thomson, N. 2001. *J: The Natural Language for Analytic Computing.* Baldock, UK: Research Studies Press.